

csv files : the program allows the input of a CSV file and parses it



Boundary value analysis

{empty file} → invalid: empty file, nothing to parse

'' '' '' '' → invalid: empty file, nothing to parse

column1, column2, column3
one , two , three → valid

column1, column2, column3
one , , three → invalid: empty value in one column

column1, column2, column3
one , two , three
one , two , three , four → invalid: extra column value

csv files can be opened from filenames : The program takes in command line interface arguments, parse the filename that was keyed in and attempts to open the file



Boundary value analysis

{empty string} → invalid: filename not specified

.csv → invalid: filename not specified

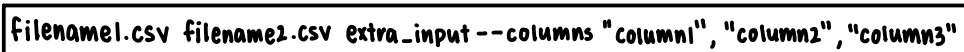
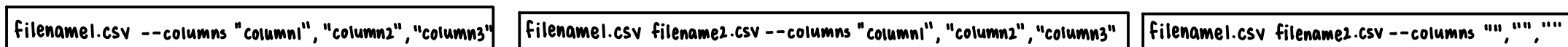
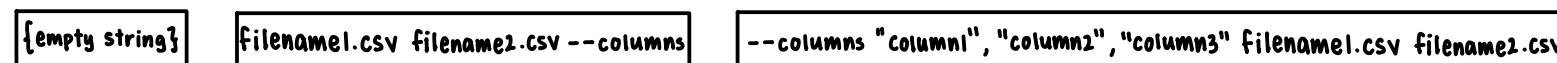
filename.csv → valid

/location/filename.csv → invalid: file should be placed in the sample folder

filename.doc → invalid: invalid file type

@!+##%{}<>*.csv → invalid: invalid characters in filename

Parse columns through command line interface arguments . This program is able to take in CLI arguments and parse the column input in a string that is separated by a comma



Boundary value analysis

{empty string} → invalid: filename and column not specified

filename1.csv filename2.csv --columns → invalid: column not specified

--columns "column1", "column2", "column3" filename1.csv filename2.csv → invalid: incorrect sequence

filename1.csv --columns "column1", "column2", "column3" → invalid: no secondary file to compare to

filename1.csv filename2.csv --columns "column1", "column2", "column3" → valid

filename1.csv filename2.csv --columns "", "", "" → invalid: column not specified

filename1.csv filename2.csv "column1", "column2", "column3" → invalid: missing "--columns"

filename1.csv filename2.csv --columns column1, column2, column3 → invalid: column names must be within ""

filename1.csv filename2.csv extra_input --columns "column1", "column2", "column3" → invalid: extra input not recognised